Piecewise Linear Regression-Based Single Image Super-Resolution via Hadamard Transform

Jingjing Luo[a,1], Xianfang Sun[b,1], Man Lung Yiu[c], Longcun Jin[a,*], Xinyi Peng[a]

[a]*School of Software Engineering, South China University of Technology, Guangzhou 510006, China*
[b]*School of Computer Science and Informatics, Cardiff University, Cardiff, UK*
[c]*Department of Computing, Hong Kong Polytechnic University, Hong Kong, China*

Abstract

Image super-resolution (SR) has extensive applications in surveillance systems, satellite imaging, medical imaging, and ultra-high definition display devices. The state-of-the-art methods for SR still incur considerable running time. In this paper, we propose a novel approach based on Hadamard pattern and tree search structure in order to reduce the running time significantly. In this approach, LR (low-resolution)-HR (high-resolution) training patch pairs are classified into different classes based on the Hadamard patterns generated from the LR training patches. The mapping relationship between the LR space and the HR space for each class is then learned and used for SR. Experimental results show that the proposed method can achieve comparable accuracy as state-of-the-art methods with much faster running speed. The dataset, pretrained models and source code can be accessed at this URL [†].

Keywords: Single image super-resolution, Hadamard transform, Decision tree
2010 MSC: 00-01, 99-00

1. Introduction

Image super-resolution (SR) is the process of recovering a visually pleasing high-resolution (HR) image from a low-resolution (LR) image. SR finds many real applications, e.g., face recognition [26, 42, 45, 63, 71], visual question answering [69], vi-

*Corresponding author. Tel.:+86 13825158906.
Email address: lcjin@scut.edu.cn (Longcun Jin)
[1]Equal contribution.
[†]$https://github.com/youyouyimu/PLRBSISRvHT$

sual speaker identification and authentication [35], object understanding [39], activity recognition [43], surveillance systems, satellite imaging, medical imaging, and ultra-high definition display devices. Most of existing methods use certain prior information to address the SR problem, especially learned priors. The interpolation-based methods [5, 20, 32, 38, 73], the reconstruction-based methods [8, 10, 16, 28, 55, 72] and the learning-based methods [7, 9, 11–15, 17, 18, 21–25, 27, 33, 34, 36, 37, 40, 41, 44, 46–48, 51–53, 56, 58–60, 62, 64–66, 70, 74–76] are three classical types of methods for single-image SR. These SR methods aim to solve natural image SR problem. There are a kind of SR methods that only deal with face images, which are called face SR (face hallucination) [29, 54]. In this paper, the former is our concern.

Learning-based SR methods divide the input LR image into patches and predict their corresponding HR patches using the mapping models that are learned from a dataset of LR-HR patch pairs. These LR-HR patch pairs are cropped from a database composed of LR-HR image pairs. Many learning algorithms have been proposed to learn the mapping models, including dictionary learning [17, 18, 22, 40, 41, 46, 56, 58–60, 65, 66, 70, 76], regression [11, 47, 48, 58, 59, 64], decision tree [24, 62], random forest [23, 25, 53] and convolutional neural network (CNN) [13, 14, 33, 34, 36, 37, 52, 57].

The advantages and limitations of the above methods are summarized below. Most of dictionary learning methods are sparse coding (SC) based, where the sparse prior can well regularize the ill-posed SR problem. However, constructing sparse dictionaries requires expensive computation. Regression-based methods can solve the SR problem by several piecewise linear regression models or a global regression model. Both of the decision tree based methods and the random forest based methods are an ensemble of piecewise linear regression models. However, the complex tree structure and a large number of trees (forest) can slow down the retrieving speed of regression models. CNN-based methods train a global non-linear regression model to describe the mapping relationship between the LR space and the HR space more accurately. The global non-linear regression model consists of a large amount of parameters, whose computating process involves heavy computational load.

More recently, regression-based methods have achieved great improvements in SR.

Linear regression models [31] have higher prediction speed than non-linear regressions models. These methods [11, 13, 14, 33, 34, 36, 37, 47, 48, 52, 58, 59, 64] learn the relationship between the LR space and the HR space, and use it to solve SR problem. Timofte et al.[58, 59] assumed that the mapping relationship between the LR space and the HR space is locally linear and therefore lots of linear regressors are learned and anchored to the feature space as a piecewise linearization. The methods [47, 48] divide the feature space into many subspaces based on antipodally invariant metrics and learn a linear regressor for each subspace. In the papers [13, 14, 33, 34, 36, 37, 52], the mapping from the LR space to the HR space is described as a deep CNN that takes an LR image as the input and outputs an HR image, which is an end to end mapping (i.e. a global non-linear regressor). The method [37] uses a generative adversarial network (GAN) for image SR, in which a perceptual loss function [30] is adopted.

SR methods based on CNN require significant amount of training time, rendering them not suitable in certain application scenarios. Some SR methods that are based on SC [58, 59] and simple functions [64] use the gradients of the LR image patches and normalized image patches, respectively, to represent image features, which also adds to the computational complexity of the training phase. In order to tackle the above limitations, we propose a Hadamard pattern-based SR method, using decision tree [24] for single-image SR.

A Hadamard matrix [3] is a square matrix that consists of +1 and -1. It is symmetric with respect to leading diagonal. Its rows (columns) are orthogonal to one another. The Hadamard transform uses a Hadamard matrix as its operator and is useful in signal-image processing [49], including signal/image coding/decoding [1, 50] and compressive sensing [67]. It could also be useful in image filtering and pattern recognition. In our paper, we perform Hadamard transform to code the LR training image patches. The coding results (called Hadamard patterns) are bases of classifying training data.

The contributions of this paper can be summarized into three aspects.

1) We propose to implement Hadamard transform on LR image patches and use the obtained Hadamard patterns to represent image features. Hadamard transform is fast because it just needs addition and subtraction without multiplication or division. This property makes feature extraction efficient.

2) We employ a variant of decision tree, a ternary decision tree, to conduct fast classification and regression.

3) The experimental results show that the proposed method can achieve comparable accuracy with much lower running time when compared to state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 defines the problem and briefly describes the related work. Section 3 presents our solution to the single-image SR problem. Section 4 analyzes the experimental results, and Section 5 concludes this paper.

2. Related Work

2.1. Problem Statement

Single image SR aims to reconstruct an HR image with high definition and fidelity from an LR image that has unsatisfactory resolution, which can be formulated as

$$\hat{X} = \uparrow Y,\ s.t.\ \hat{X} \approx X \tag{1}$$

where Y is the LR input image, $\hat{X}$ is the upscaled output image, X is the original HR image and $\uparrow$ is an upsampling operator. In the training phase, X is known. This formula implies that the upsampling mapping models are trained to describe the relationship between the LR space and the HR space as accurately as possible. In the testing phase, X is unknown. This formula implies that the learned mapping models generate an HR output from an LR input. The predicted HR output and the HR image generated from the same imaging model are as similar as possible.

In the literature, the following transformation is usually used to describe the real imaging process of LR images

$$Y = \downarrow BX + n \tag{2}$$

where $\downarrow$ is the downsampling operator, B is the blurring operator, and n is the additive noise. Most SR methods solve this problem at a patch level.

2.2. State of the Art

Recently, the single-image SR problem has been investigated extensively. Interpolation-based methods [5, 20, 32, 38, 73], reconstruction-based methods [8, 10, 16, 28, 55, 72] and learning-based methods [7, 9, 11–15, 17, 18, 21–25, 27, 33, 34, 36, 37, 40, 41, 44, 46–48, 51–53, 56, 58–60, 62, 64–66, 70, 74–76] are three common types of single image SR methods. Interpolation-based methods produce blurry edges in the reconstructed images. Reconstruction-based methods fail to restore the novel details of HR images, especially when a large upsampling factor is employed. Learning-based single-image SR methods have attracted much research interests. In the following subsections, we will first introduce the SR problem and then focus on the related work of learning-based methods.

SC based methods [46, 58, 59, 65, 70] decompose an input LR image patch into a sparse linear combination of the atoms in the LR dictionary. The target HR patch is generated by corresponding atoms in the HR dictionary with the same representation weights. Zeyde et al. [70] applied the orthogonal matching pursuit (OMP) and reduced the dimensionality of feature vectors by principal component analysis (PCA), which improves both the running time and the accuracy. Peleg and Elad [46] suggested data clustering and cascading several levels of their proposed SR model, which is like a simple feedforward neural network and reduces the algorithm complexity. Timofteet et al. [58] proposed the anchored neighbor regression (ANR) method. The ANR method is much faster than Zeyde's method [70]. The same researchers proposed an A+ method [59], which is an improved version of ANR. The A+ method uses the neighborhood LR-HR patch pairs to learn the regression model rather than the neighborhood dictionary atoms. The ANR and A+ methods change the way sparse-coded dictionaries are used, where they search the nearest neighbors for each dictionary atom and learn the regression model for each atom using its nearest neighbors. Dong et al. [13, 14] proposed an SR method based on convolutional neural network (SRCNN). SRCNN consists of three layers. Training SRCNN [13] takes roughly 3 days to converge. Lai et al. [36] proposed the Laplacian Pyramid Super-Resolution Network (LapSRN) based on a cascade of CNNs. LapSRN progressively generates the sub-band residuals of HR output images in a coarse-to-fine scheme. The generated residuals at each level are

used to reconstruct the target HR output by upsampling and addition operations. The training time of LapSRN on a Titan X GPU is about 3 days.

Decision tree can be used to support classification and regression. Huang et al. [24] presented a decision tree-based method for single image SR, which uses all the training data to initialize the root node and generates the hierarchical decision trees. This algorithm fuses regression models that come from the same tree to improve accuracy. Random forest-based SR methods [23, 25, 53] divide a training dataset into many subsets, each of which is used to train a decision tree. When the training is done, each leaf node of a decision tree corresponds to a learned regression model. In the testing stage, the input LR patch is passed to a leaf node and converted to an HR patch using the corresponding regression model. The learned decision trees form the SR random forest. Huang et al. [25] grouped a training dataset into four subsets depending on the structure of pixels in the interpolated images. An SR decision tree is learned for each subset.

The ANR and A+ methods search nearest neighbors for each dictionary atom, which is equivalent to grouping dictionary atoms into different classes. The number of classes equals the number of dictionary atoms. So aforementioned learning-based SR methods [9, 23–25, 47, 48, 58, 59, 64] divide the training data into different classes and learn a regression model for each class. The searching time of the methods [9, 58, 59, 64] is linear to the number of classes, and that of methods [23–25, 53] is approximately linear to the depth of the tree, i.e., the logarithm of the number of leaf nodes. This property of decision tree makes it possible to achieve fast SR. Two other methods [47, 48] use a Spherical Hashing (SpH) [19] algorithm to search a regressor. They rely on a parallel implementation to deal with the sparse dictionaries with a large amount of atoms. Besides Spherical Hashing algorithm, the emerging cross-media hashing algorithm [68] may be used to search a proper regressor in a learning-based SR method.

In addition, the way that learning-based methods [46, 58, 59, 64, 65, 70] extract image feature can add to computational complexity, and the CNN-based methods take significant training time.

We propose to use Hadamard patterns as image features for SR because it can

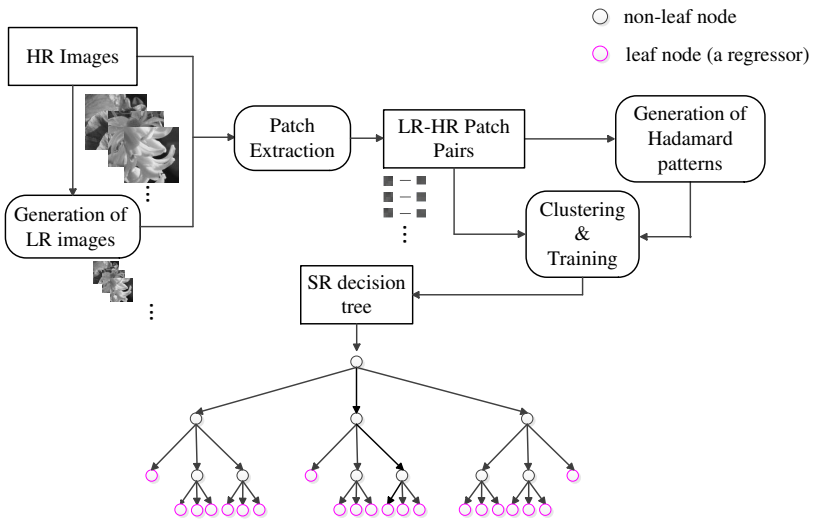


Figure 1: The flowchart explaining the steps used to construct a training dataset of LR-HR patch pairs and learn the SR decision tree from this training dataset. We divide the training data into different groups based on the generated Hadamard patterns of LR patches using a coarse-to-fine strategy, and then learn a mapping model for each group. We cluster the training data and at the same time learn a SR decision tree.

represent typical LR image patterns. Also, Hadamard pattern is simpler to compute than image gradient [58, 59], which is not computationally demanding and only needs small memory. Decision tree is used in our method to achieve fast classification and regression.

3. The Proposed Method

In this section we present our method for fast regression-based SR. Our proposed method is based on the idea of piecewise linear regression presented in [47, 48]. It employs an ensemble of piecewise linear mapping models (piecewise linear regressors) anchored to certain leaf nodes and searches the appropriate mapping model through a ternary search algorithm. We train and select the mapping models similarly, as shown in the upcoming sections.

An overview of our algorithm is shown in Fig. 1. It explains how to construct a training dataset of LR-HR image patch pairs. Based on the generated Hadamard patterns of the LR training image patches, an SR decision tree is learned, where the

LR-HR image patch pairs cluster into different classes on the leaf nodes. A linear mapping model is learned for each leaf node using the training data arriving at this node. Since all the data are classified into different classes based on the SR decision tree, and each class is modelled by a linear mapping model, the decision tree represents a piecewise linear system.

3.1. Linear Regression Framework

Some CNN-based SR methods [13, 14, 33, 34, 36, 37, 52] choose a non-linear regression scheme, which leads to prohibitive complexity in training. In testing phase, the complicated and deep network structure (a global regression model) may affect running time.

In our method, we choose a simple regression scheme, i.e. piecewise linear regression. Considering one linear regressor (mapping model), we just need to compute the linear weighted sum of the input LR pixels (they compose an LR image patch) for a desired output HR patch. We split the feature space into many subspaces and obtain a better piecewise linear regression system that consists of many linear mapping models. In the training phase, the training data are grouped, and during the testing phase the proper mapping model is selected for each input LR patch.

For the SR problem discussed in this paper, regression is used to reconstruct HR patches from the LR input patches using a set of linear mapping models that conform a piecewise linear regression system. The process can be formulated as:

$$h^r = lM_q \tag{3}$$

where l is a row vector, representing a vectorized input LR image patch. The reconstructed HR image patch of l is represented by a row vector h^r. M_q (q=1,2,...,D) is a coefficient matrix, representing the q^{th} mapping model and D is the total number of the learned mapping models.

3.2. Learning the SR decision tree

In this paper, we propose an SR method using Hadamard patterns. The Hadamard patterns are obtained by Hadamard transform that uses Hadamard matrix as the trans-

formational matrix. The calculation of Hadamard pattern is simple because it just involves addition and subtraction. Our method uses the Hadamard patterns of LR image patches to describe their image features, and clusters training data based on them. For each class, we train the mapping model M_q from the LR space to the HR space using the training data that belong to it.

The concept of decision tree is first proposed by Breiman et al. [6], which is often used in data mining. The decision tree has a tree structure, where a node with two or more child nodes is called a non-leaf node and a node without a child node is called a leaf node.

In this paper, we employ the ternary decision tree for image SR, where each non-leaf node has three child nodes. We have conducted experiments on both the traditional binary decision tree and the ternary decision tree. Experimental results show that the ternary decision tree can result in better accuracy than the binary decision tree.

In the training stage, the root node of the SR decision tree is initialized with all the training data. Then threshold parameters are learnt to determine how each non-leaf node with sufficient training data splits the training data into its left, middle and right child nodes. Finally, each leaf node holds some training data, which are used to learn a mapping model.

3.2.1. The Extraction of Training Data

We extract LR-HR patch pairs from the training image dataset, which consists of LR-HR image pairs. The HR training images I_h are directly downsampled by bicubic interpolation to generate corresponding LR training images I_l. The extracted LR-HR patch pairs (l_j, h_j) are used for training, where $l_j \in R^{1\times 16}$ is a vectorized LR image patch and h_j is a vectorized HR image patch.

3.2.2. Feature Representation

The selection and extraction of LR image feature will affect the quality of HR output and the running time. The simple function method [64] uses the normalized LR patch as image feature. The methods [58, 59, 70] use the first-order and second-order gradients of an LR image patch as its image features, which adds to computational

complexity. In this paper, we perform Hadamard transform on vectorized LR image patches to calculate their Hadamard patterns. The implementation of transform is fast, which saves the running time. Eq. (4) shows how to calculate the Hadamard pattern p_i for a vectorized LR image patch l_i.

$$p_i = l_i Q_{15} \tag{4}$$

where Q_{15} is the reduced Hadamard matrix described below.

$$Q_{16} = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1
\end{bmatrix} \tag{5}$$

A 16-order Hadamard matrix Q_{16} is used in our paper. The Eq. (5) shows the Hadamard matrix Q_{16}. Q_{16} has all its rows (columns) orthogonal to each other and any two rows (columns) of Q_{16} differ in half their elements. The Hadamard matrix with order of a power of 2 (e.g. Q_{16}) can be constructed through the following recursive method (6). It can be realized by a built-in MATLAB function (i.e. hadamard()).

$$Q_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad Q_{2^{k+1}} = \begin{bmatrix} Q_{2^k} & Q_{2^k} \\ Q_{2^k} & -Q_{2^k} \end{bmatrix} (k \geq 1) \tag{6}$$

In our method, we extract square patches and vectorize them. Each patch is represented by a row vector. According to Eq. (6), the order of Hadamard matrix is 2, 4, 8, 16, 32, 64, ..., $2^k (k \geq 1)$. In order to perform Hadamard transform on vector-

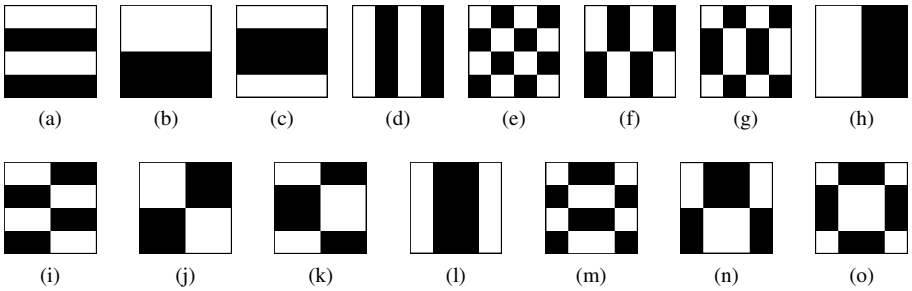


Figure 2: The visualization of Hadamard matrix Q_{15}. Each column of the Hadamard matrix Q_{15} is treated as a convolution filter of 4×4.

ized square patches (see Eq. (4)), the side length of square patches needs to be an even number.

We set the size of an LR image patch to 4×4 and use a 16-order Hadamard matrix Q_{16} (see Eq. (5)) in our method. When the size of an LR image patch is 2×2, its corresponding Hadamard matrix is Q_4. The small patch size will result in capturing less image information. When the size of an LR image patch is 8×8, its corresponding Hadamard matrix is Q_{64}. The accuracy of adopting Q_{64} is worse than that of adopting Q_{16}. Besides, the order of Q_{64} is bigger. We need to compute more parameters in our method, which adds to the computation load and may cause over-fitting.

Each column of Q_{16} is equivalent to a convolution filter, which can be used to obtain different statistics of the image features of an LR image patch. The first column of the Q_{16} is all 1. This filter just calculates the sum of all pixels in an LR image patch. It can not reflect the texture characteristics (texture diversity) of an LR image patch. So we delete the first column of Q_{16} to get a new matrix Q_{15}. Later, the so-called Hadamard matrix refers to the new matrix $Q_{15} \in R^{16\times15}$.

Each column of Q_{15} can be reshaped as a 4×4 matrix and visualized. Fig. 2 is the visualization of Hadamard matrix Q_{15}, where the white pixel represents +1 and the black -1. The subgraph (a)-(o) represents the $1^{st} - 15^{th}$ column of Q_{15} respectively.

The big block of black (white) blocks implies low frequency signal and small size implies high frequency signal. Small size is more likely related to noise. In consideration of this, we set the sequence Seq = [2 8 3 12 10 1 4 11 14 6 9 15 7 13 5]. Following

this determined sequence, we perform 15 round of splittings to group training data into different classes using the corresponding column of Hadamard patterns of LR training data.

3.2.3. Clustering and Training

All the extracted LR image patches form a matrix $L \in R^{N\times 16}$ by stacking the row vectors $l_i (i = 1, ..., N)$, where N is the number of extracted pairs, and the corresponding HR image patches form a matrix $H \in R^{N\times s^2}$ (s is the upscaling factor). The Hadamard pattern $P \in R^{N\times 15}$ can be calculated by the following Eq. (7).

$$P = LQ_{15} \tag{7}$$

In the process of learning an SR decision tree, the training data are split from non-leaf nodes into leaf nodes to perform clustering on training data. Once the clustering is done, each cluster is used to learn a mapping model. We initiate the root node with all the training data. According to the determined sequence Seq, in the first round of splitting, we use the 2^{nd} (Seq[1]) column of the generated Hadamard patterns to group training data into three classes based on two learnt thresholds described below. By now, the SR decision tree has one root node and three child nodes. Then the three classes go on the second round of splitting separately. During the second round of splitting, we use the 8^{th} column (Seq[2]) of the Hadamard patterns corresponding to each class to perform further classification. The splitting goes on round after round and there are at most 15 round of splittings. In the $k^{th} (k = 1, ..., 15)$ round of splitting, how many times we perform classification depends on the current number of non-leaf nodes. For each non-leaf node, its training data are partitioned according to the $Seq[k]^{th}$ column of its Hadamard patterns. The SR decision tree is progressively constructed by partitioning the training data from non-leaf nodes into leaf nodes. When we are performing the k^{th} round of splitting, we are constructing the k^{th} layer of the SR decision tree. The depth of the learned SR decision tree is at most 16 because Q_{15} has 15 columns and so the generated Hadamard patterns has 15 columns. Here, we consider that the root node has depth 1. When we have finished the 15^{th} round of splitting, we mark the

generated child nodes as leaf nodes.

In the splitting at the non-leaf node j with N_j training patch pairs (i.e., L_j and H_j), we assume that it has sufficient training data. We generate Hadamard patterns P_j (see Eq. (7)) for training data in current node j. We assume that this splitting is in the $k^{th}(k = 1, ..., 15)$ round of splitting. We use the $Seq[k]^{th}$ column of P_j to group L_j and H_j. First we sort the $Seq[k]^{th}$ column of P_j in ascending order to get P^s as shown in Eq. (8). Then the threshold values v_1 and v_2 are obtained from $P^s \in R^{N_j}$ by Eq. (9)

$$P^s = Sort(P_j(:, Seq[k]^{th})) \tag{8}$$

$$v_1 = P^s(\lceil (1-v)N_j/2 \rceil, 1), \quad v_2 = P^s(\lfloor (1+v)N_j/2 \rfloor, 1) \tag{9}$$

where $v \in (0, 1)$ controls the position we select threshold values. It is the constraint we employ to restrict the relative training data size of three child nodes.

The training data that their Hadamard pattern values in the $Seq[k]^{th}$ column are smaller than v_1 will be passed to the left child node, and greater than v_2 to the right child node. The rest of the training data will be passed to the middle child node. Then a set of parameters $\beta_j = [k, ind_l, ind_m, ind_r, v_1, v_2]$ are stored in this non-leaf node. The parameter k is the depth of this node (i.e. this node's training data are classified in which round of splitting). ind_l, ind_m and ind_r are indices that point to this node's left, middle and right child node respectively. v_1 and v_2 are the learned threshold values. v_1 is the smaller one.

The ternary splitting should fulfill Eq. (10).

$$min(N_l, N_m, N_r) \geq min\_num \tag{10}$$

where N_l, N_m and N_r are the training data sizes of the left, middle and right child node respectively, and min_num represents the minimum size of training samples in one leaf node.

After one splitting, if the number of training samples in one child node is less than the minimum min_num we have determined, then this splitting is invalid, and the

current node is marked as a leaf node. Before we begin a new splitting, we also check whether the number of training samples in current node is less than $3 * min\_num$.

If it is, then the current node is marked as a leaf node because splitting this node will result in at least one child node whose training samples are less than min_num. For each leaf node, we learn a mapping model from the LR space to the HR space using the training data arrived here and just store the index to this mapping model in this leaf node.

A non-leaf node in the learned SR decision tree stores its depth (i.e. this node's training data are classified in which round of splitting) in the tree, the learned threshold values and the indices that point to its child nodes. The learned threshold values are used for classification. The training and testing data are partitioned into this non-leaf node's left, middle or right child node according to a comparison result between the learned threshold values and their Hadamard patterns. A leaf node in the learned SR decision tree stores the index that points to a mapping model. Let M_q be the mapping model that is learned using training data arrived at leaf node q.

For each leaf node q, a mapping model is calculated using the least squares method to solve Eq. (11) with the constraint that the sum of each column of M_q is 1.

$$H_q = L_q M_q \tag{11}$$

where M_q is the regression matrix of the q^{th} leaf node, H_q and L_q are the training data pairs arriving at leaf node q. The clustering and training scheme is summarized in Algorithm 1.

3.3. Time Complexity Analysis

Let N be the total number of the LR-HR patch pairs in the training dataset, and L be the number of layers in the decision tree. The time complexity of training the decision tree is $O(N \cdot L)$, because N Hadamard patterns need to be computed in each layer. In the proposed method a constant (at most 15) layers exist in the decision tree, so the time complexity of training is $O(N)$.

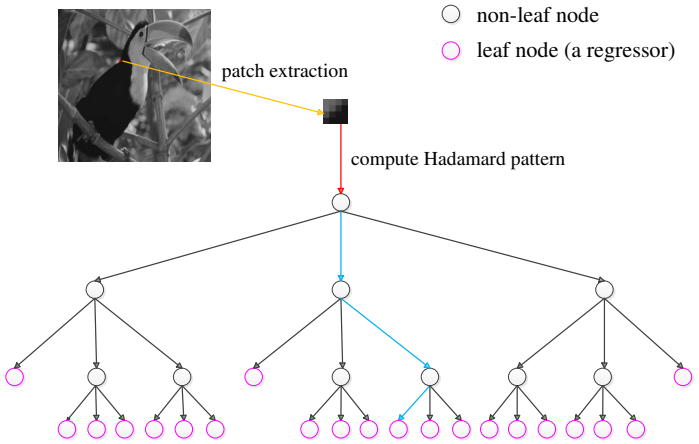


Figure 3: The flowchart explaining the searching process. Each time we extract one LR patch form the input LR image and compute its Hadamard pattern. Then this generated Hadamard pattern is compared with the learned thresholds in the obtained SR decision tree until arrives at a leaf node. The proper mapping model is found.

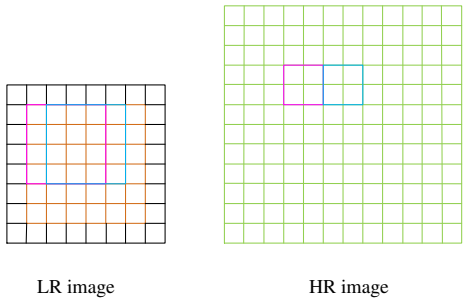


Figure 4: The position relationship between an input LR patch and its predicted HR patch. In this figure, each small square represents a pixel. The black squares in the LR image are padding and the yellow squares are LR pixels. The right one is the predicted HR image with the upscaling factor $s = 2$. Obtaining all the predicted HR patches, we crop the border to get the final HR output.

3.4. SR Scheme

For regression-based SR methods, searching proper mapping models in the stage of performing SR is time-critical. In our method, we perform a ternary search, which yields state-of-the-art running time. The searching process is shown in Fig. 3. When we search a mapping model for an input LR image patch, we need to go through at most 15 (a constant) number of layers of the decision tree. Each layer involves only one Hadamard pattern computation and two comparisons, so the time complexity of traversing the decision tree is $O(1)$.

We use a sliding window of 4×4 to extract a patch from the input LR image in a raster-scan order and calculate its Hadamard pattern. Each time the sliding window moves one pixel. According to the current depth, we compare the $Seq[depth]^{th}$ column of the generated Hadamard pattern with the thresholds corresponding to current node and pass this test patch to next node until it arrives at a leaf node. The corresponding mapping model is then used to map the LR image patch to the target HR output. All the mapped HR patches form the predicted HR output. The extracted patches from input LR image have overlapping regions, and the predicted HR image patches have no overlapping regions. So we need not to average the overlapping regions for the target output in the testing stage. As seen in Fig. 4, when the upscaling factor is 2, the sliding window in input LR image moves one pixel and the sliding window in predicted HR image moves two pixels.

4. Experimental Results and Analysis

In this section, we show experimental results and analyse them. In order to demonstrate effectiveness and efficiency of the proposed algorithm, we employ a common training image dataset and two testing datasets. We use the training image dataset [65] that contains 91 natural images. For testing image datasets, we use Set5 [4] (5 images) and Set 14 [70] (14 images). The testing images are not included in the training image dataset. We adopt peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) [61] and running time to measure the performance for all methods.

Following the experimental setting of other papers [14, 24, 36, 53, 58, 59], we extract LR training data from the standard image dataset without Gaussian blurring on these training images. For generating the testing images, however, to imitate the real imaging process, all the images in Set5 and Set14 are first blurred by a 5×5 Gaussian filter with standard deviation 1.6 and then downsampled by bicubic interpolation with scaling factor of 2 to generate the corresponding LR testing images. The size of an LR image patch is 4×4 and the size of an HR image patch is s×s (s is the upscaling factor). The proposed method only deals with the luminance channel in YCrCb color space and the chromatic components are enlarged to the desired size with bicubic interpolation.

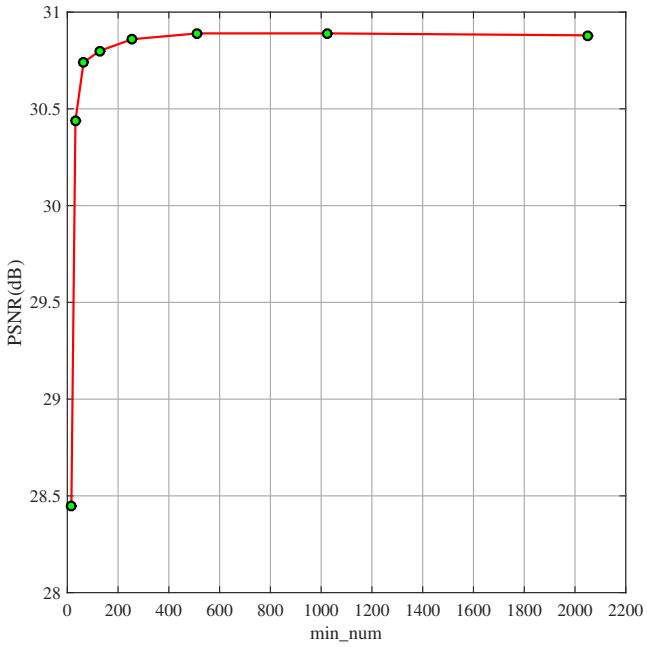
We use the LR testing images in the aforementioned conditions to measure all methods [14, 24, 36, 53, 58, 59].

4.1. Experimental Settings

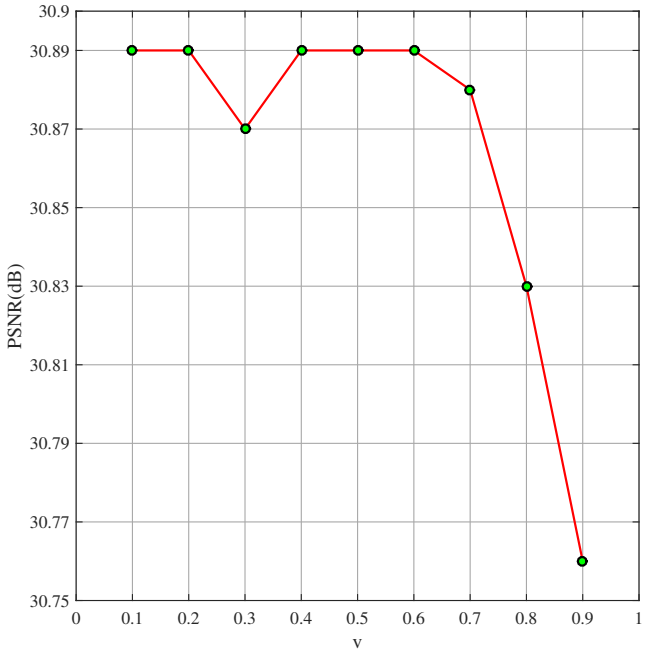
The quality of the proposed method is related to the quality of the mapping models. It is obvious that the quality of mapping models can be improved by using more training data. More training data can make the relationship between LR and HR image patches be described more accurately. Fig. 5 (a) illustrates the relationship between the minimum size of training data in a leaf node min_num and the average PSNR of the reconstructed images in Set5 when the upscaling factor is 2 and v is 0.4. We vary min_num from 16 to 2048 and each time min_num doubles. It can be seen from Fig. 5 (a) that the average PSNR of these testing images is improved by more than 2.0dB. As min_num is larger than 512, the accuracy converges. In the following experiments, min_num is set to 512.

In our proposed method, we use the v value to restrict the relative training data size of three child nodes, as stated in Eq. (9) and Eq. (10). The average PSNR of the reconstructed images in Set5 with respect to different v are shown in Fig. 5 (b). A larger or smaller v gives a more tight constraint on the training data size of child nodes, which results in longer training time. Considering both the training effectiveness and the training time, in the following experiments, we set v to 0.5.

Our computing platform is an Intel Core i7-4790 CPU 3.60 GHZ 4 core processor with 16GB RAM.

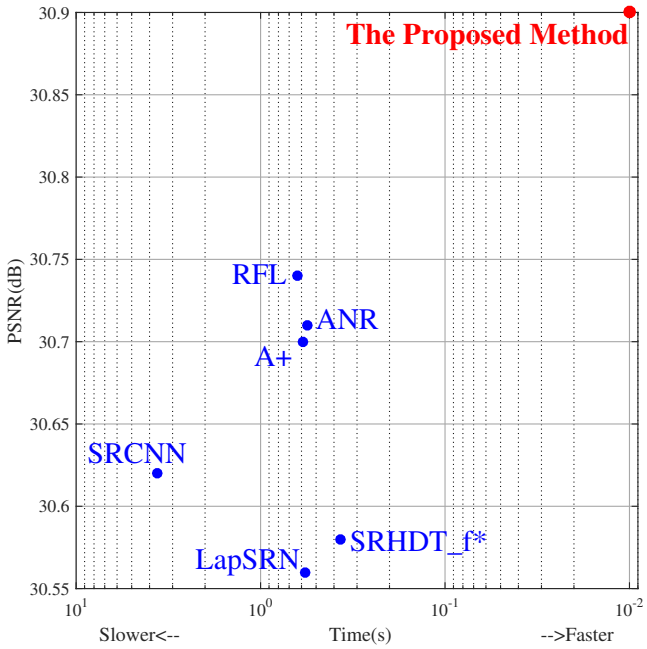


(a)

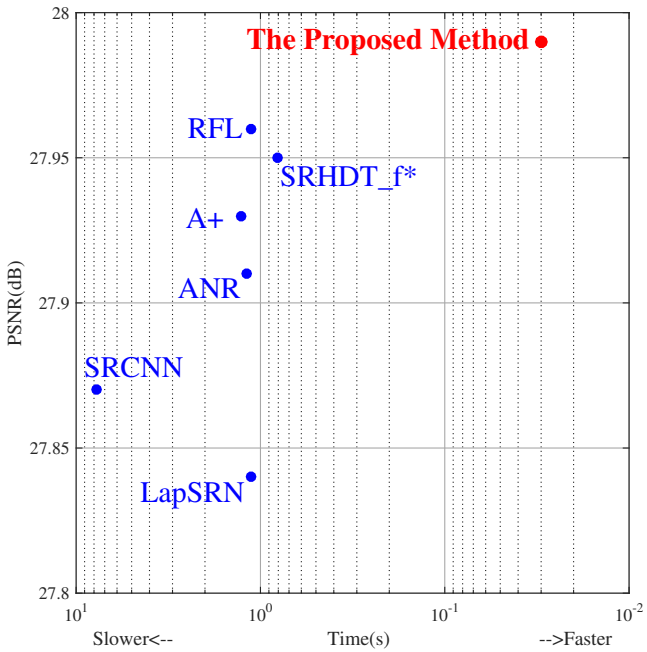


(b)

Figure 5: Upscaling factor=2:(a) the relationship between min_num and the average PSNR of the reconstructed images in Set5 when upscaling factor is 2, with v =0.4; (b) the relationship between v and the average PSNR of the reconstructed images in Set5 with min_num=512.



(a)



(b)

Figure 6: Speed and accuracy trade-off. The results of (a) and (b) are evaluated on Set5 and Set14 respectively with the upscaling factor 2. The proposed method generates SR images efficiently and accurately.

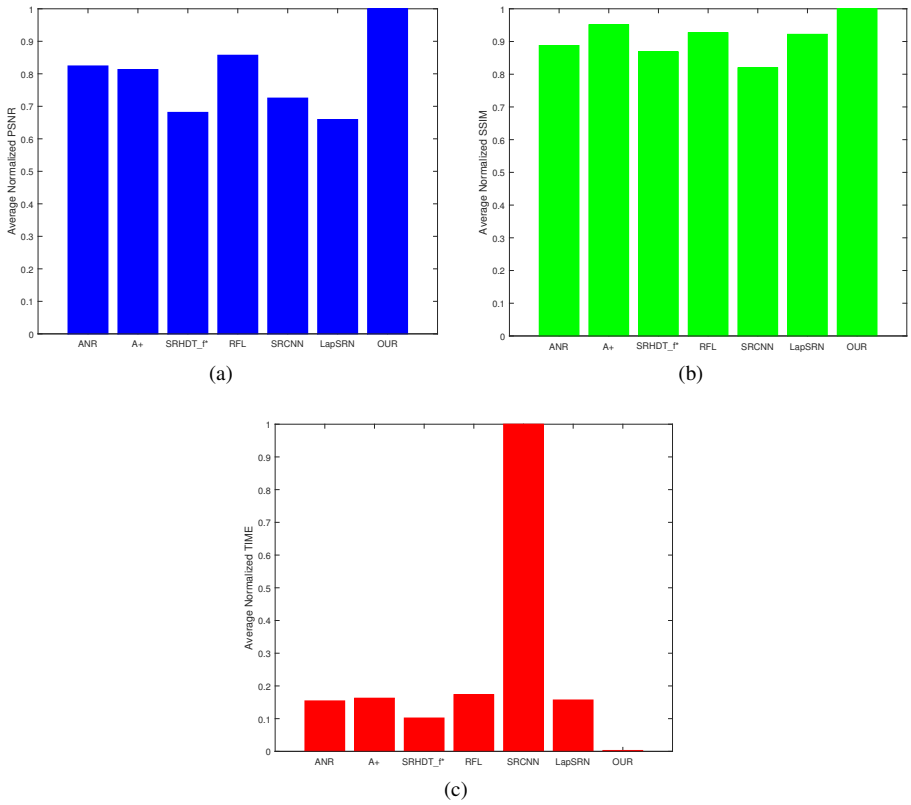


Figure 7: The normalized average PSNR, SSIM and running time by different methods for scaling factor 2 on Set5.

4.2. Comparison With State-of-the-Art Methods

We compare our method with some well-known single image SR methods, such as ANR [58], A+ [59] ,SRHDT_f* [24], RFL [53], SRCNN [14] and LapSRN [36] methods. We have used their published implementations. Fig. 6 shows the trade-off between running time and accuracy clearly. The proposed method performs best.

In Fig. 7, we show the normalized average PSNR, SSIM and running time by different methods on Set5. In Table 1, we provide a detailed quantitative evaluation on Set14. Among them, the Bicubic, ANR, A+, SRCNN, LapSRN methods are implemented in MATLAB 2015a. The RFL and the proposed methods are implemented based on mixed programming of C++ and MATLAB. For the SR method in [24], we

use their proposed hierarchical framework with fused regression models (SRHDT_f) for comparison. SRHDT_f* is implemented in C++ and applies the standard 91 training images for training, but its running time is still slower than ours as shown in Fig. 7 and Table 1. For SRCNN method, its published code is slower than the one used in their paper. However their reported times for SRCNN are slower than those of A+ in their benchmark [14] and our running times are faster than those of A+ as shown in Fig. 7 and Table 1. For LapSRN method, we only test its accuracy and running time on a CPU platform. They applied GPU acceleration in their published paper [36].

As can be seen in Fig. 7 and Table 1, comparing with the state-of-the-arts, our method can achieve comparable accuracy with much less running time. In addition, visual comparison in Fig. 8 - 9 shows that our method can better preserve sharp edges and restore more detailed information.

4.2.1. SR methods based on sparse coding

The proposed method, ANR method and A+ method use the same training image dataset [65]. As shown in Fig. 7, the average PSNR of our method is higher than both the average PSNR achieved by ANR method and the average PSNR of A+ method. Besides, our method leads in the average running time. In addition, our method can obtain higher average SSIM. The A+ method improves ANR method. As shown in Fig. 7 and Table 1, our method outperforms ANR and A+ in running time while being competitive in quality.

When searching the nearest neighbor dictionary atom for an input LR feature, the ANR and A+ methods calculate the correlation between the input LR feature and all dictionary atoms to find the dictionary atom with maximum correlation. The searching time is linear to the size of this dictionary. The searching process of our method is a ternary search. In our method, the learned SR decision tree has at most 16 layers (the depth of the root node is 1). So we only need at most 15 comparisons until the desired leaf node is found. In the ANR and A+ methods, there are 1024 atoms in the dictionary. They need to find the nearest neighbor atom from them. Obviously, we are the best-performer in the trade-off between quality and running time, confirming the best appropriateness of practical application.

Table 1
The PSNR, SSIM and running time by different methods for scaling factor 2 on Set14.

Images	Measures	Methods							
		Bicubic(M)	ANR [58]	A+ [59]	SRHDT_f* [24]	RFL [53]	SRCNN [13]	LapSRN [36]	OUR
baboon	PSNR	23.17	23.47	**23.48**	**23.48**	23.47	**23.48**	**23.48**	23.44
	SSIM	0.5299	0.5624	0.5628	0.5625	0.5618	0.5626	**0.5639**	0.5621
	TIME	0	1.11	1.20	1.13	1.24	8.14	1.10	**0.03**
barbara	PSNR	26.11	26.46	**26.47**	**26.47**	26.45	26.46	26.44	26.40
	SSIM	0.7485	0.7691	**0.7698**	0.7637	0.7679	0.7687	0.7695	0.7672
	TIME	0	1.86	2.03	1.29	1.93	14.53	2.00	**0.05**
bridge	PSNR	24.27	24.74	24.77	**24.79**	24.78	24.77	24.74	**24.79**
	SSIM	0.6322	0.6685	0.6695	**0.6713**	0.6678	0.6700	0.6699	0.6686
	TIME	0	1.18	1.32	1.38	1.35	9.06	1.25	**0.03**
coastguard	PSNR	26.49	27.04	27.08	27.04	27.05	27.04	27.06	**27.09**
	SSIM	0.6180	0.6557	0.6581	0.6398	0.6523	0.6566	**0.6596**	0.6422
	TIME	0	0.51	0.55	0.37	0.56	2.41	0.51	**0.01**
comic	PSNR	23.00	23.59	23.60	**23.67**	23.59	23.57	23.56	23.50
	SSIM	0.6943	0.7335	**0.7356**	0.7335	0.7342	0.7314	0.7343	0.7288
	TIME	0	0.46	0.49	0.47	0.55	2.43	0.58	**0.01**
face	PSNR	32.65	33.10	**33.11**	33.02	33.04	33.06	**33.11**	33.04
	SSIM	0.7920	0.8063	0.8071	0.8033	0.8073	0.8088	**0.8115**	0.8085
	TIME	0	0.40	0.42	0.21	0.45	1.97	0.44	**0.01**
flowers	PSNR	27.04	27.71	27.73	**27.78**	27.70	27.68	27.63	27.64
	SSIM	0.7994	0.8246	**0.8257**	0.8221	0.8253	0.8235	0.8252	0.8218
	TIME	0	0.93	1.00	0.76	0.94	6.16	0.91	**0.02**
foreman	PSNR	30.25	31.30	31.33	31.38	31.59	31.17	31.21	**32.29**
	SSIM	0.8921	0.9071	0.9096	0.9127	0.9099	0.9051	0.9093	**0.9160**
	TIME	0	0.52	0.56	0.29	0.54	2.41	0.49	**0.01**
lenna	PSNR	31.33	32.01	32.00	31.98	31.99	31.92	31.92	**32.02**
	SSIM	0.8531	0.8671	0.8685	0.8639	0.8667	0.8654	**0.8686**	0.8678
	TIME	0	1.43	1.52	0.68	1.22	9.16	1.26	**0.03**
man	PSNR	26.87	27.38	27.40	**27.41**	27.39	27.35	27.32	27.36
	SSIM	0.7429	0.7671	**0.7678**	0.7644	0.7661	0.7667	0.7674	0.7663
	TIME	0	1.43	1.53	1.08	1.34	9.03	1.26	**0.03**
monarch	PSNR	29.07	29.87	29.90	**29.97**	29.95	29.81	29.71	29.95
	SSIM	0.9174	0.9294	0.9309	0.9291	0.9298	0.9277	0.9304	**0.9311**
	TIME	0	2.12	2.27	1.07	1.80	13.77	1.86	**0.05**
pepper	PSNR	31.95	32.66	32.70	32.70	32.79	32.54	32.53	**32.91**
	SSIM	0.8629	0.8741	0.8763	0.8744	0.8764	0.8728	0.8765	**0.8786**
	TIME	0	1.43	1.51	0.66	1.24	9.08	1.26	**0.03**
ppt3	PSNR	23.64	24.36	24.43	**24.52**	24.51	24.32	24.20	24.35
	SSIM	0.8772	0.8994	0.9058	0.9055	**0.9066**	0.9012	0.9019	0.9037
	TIME	0	1.87	2.00	0.85	1.54	12.49	1.63	**0.04**
zebra	PSNR	26.15	27.07	27.06	27.08	27.09	26.96	26.89	**27.13**
	SSIM	0.7812	0.8140	0.8150	0.8133	0.8151	0.8125	0.8146	**0.8161**
	TIME	0	1.23	1.31	1.12	1.16	7.70	1.11	**0.02**
Average	PSNR	27.30	27.91	27.93	27.95	27.96	27.87	27.84	**27.99**
	SSIM	0.7673	0.7913	**0.7930**	0.7900	0.7919	0.7909	**0.7930**	0.7913
	TIME	0	1.18	1.27	0.81	1.13	7.74	1.12	**0.03**

4.2.2. SR methods based on decision tree and random forest

For upscaling factor of 2, the average PSNR of our method is higher than that of SRHDT_f* and our average running time is faster as shown in Fig. 7. The average results of Table 1 show that we are consistently better than SRHDT_f* in average PSNR. Besides, our method has a faster running speed. The speed-up with respect it is $\times 27$. The hierarchical structure and fusing relevant predicted results within the same decision tree result in its longer running time.

The RFL method learns multiple trees. During inference, RFL has to average pre-

dictions over all trees. The proposed method just has one SR decision tree and has no averaging operations. We are competitive in terms of PSNR and SSIM when compared with RFL. Beyond that, we are much faster.

The tree structure in SRHDT_f* is more complex than that of ours and RFL method has more trees than the proposed method. They fail to balance the accuracy and running time due to higher computational complexity.

4.2.3. SR methods based on CNN

The SRCNN method [14] is trained on two different training image datasets respectively, one is the training dataset contains 91 images and another is the ILSVRC 2013 ImageNet training dataset contains 395, 909 images. The SRCNN performs better on the second training dataset. So the SRCNN method used in this paper is trained on the ImageNet.

As shown in Fig. 7 and Table 1, the average PSNR obtained with our method is higher than that of SRCNN. Besides, our method is highly competitive in terms of running time. Compared with SRCNN method, our method clearly outperforms it in both running time and average quality (PSNR). Furthermore, our training time is less than 15 seconds and the training time of the SRCNN method is roughly 3 days. It can be seen from Table 1, the speed-up with regard to SRCNN is $\times 258$.

The LapSRN method constructs its network based on the Laplacian pyramind framework. The HR output image is reconstructed progressively. The LapSRN method uses the 91 images from [65] and 200 images from the training dataset of Berkeley Segmentation Dataset [2] to train its model. We show the objective evaluation in Fig. 7 and Table 1. The PSNR obtained with our method is higher than that of LapSRN and our running speed is much faster. As shown in Table 1, the speed up by using our method is $\times 37.33$.

The structure of our decision tree with Hadamard patterns is much simpler than that of a deep CNN, but it resulted in similar or even better accuracy. The reason is that Hadamard patterns cover almost all the possible patterns an LR image patch can have and division of the whole feature space into piecewise linear subspaces based on Hadamard patterns can effectively approximate the non-linear feature space. So, our

approach can reach the results a complex deep CNN may produce.

Therefore, when quality of image with a fast processing speed is the priority, our proposed method can be a better candidate than aforementioned methods in real applications.

5. Conclusion

This paper proposed a Hadamard pattern-based SR method for single-image SR problem, which focuses on reducing the running time dramatically while preserving the accuracy. Our method performs Hadamard transform on LR image patches and use the generated Hadamard patterns to describe their image features. The calculation of Hadamard pattern is simple because it just involves addition and subtraction. Moreover, the use of ternary decision tree speeds up our method further.

6. Acknowledgments

This work was partly supported by National Science Foundation of China Grant 61300135, Pearl River Technology Nova Project Grant 201710010020, the Fundamental Research Funds for the Central Universities Grant x2rjD2153900, Hong Kong Scholars Program Grant XJ2014058, the Open Project Program of the State Key Lab of CAD&CG Grant A1619.

The authors would like to thank the editors and anonymous reviewers for their comments and suggestions.

References

[1] Anshi, C., Di, L., Renzhong, Z., 1993. A research on fast hadamard transform digital systems, in: IEEE Region 10 Conference on TENCON'93. Proceedings. Computer, Communication, Control and Power Engineering, pp. 541–545. doi:10.1109/TENCON.1993.328044.

[2] Arbelaez, P., Maire, M., Fowlkes, C., Malik, J., 2011. Contour detection and hierarchical image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 898–916. doi:10.1109/TPAMI.2010.161.

[3] Bell, D., 1966. Walsh functions and hadamard matrixes. Electronics Letters 2, 340–341. doi:10.1049/el:19660286.

[4] Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L., 2012. Low-complexity single-image super-resolution based on nonnegative neighbor embedding, in: British Machine Vision Conference (BMVC), BMVA press. doi:10.5244/C.26.135.

[5] Blu, T., Thévenaz, P., Unser, M., 2004. Linear interpolation revitalized. IEEE Transactions on Image Processing 13, 710–719. doi:10.1109/TIP.2004.826093.

[6] Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A., 1984. Classification and regression trees. CRC press. doi:10.1007/978-3-319-03629-8_10.

[7] Cao, F., Li, K., 2018. A new method for image super-resolution with multi-channel constraints. Knowledge-Based Systems 146, 118 – 128. doi:10.1016/j.knosys.2018.01.034.

[8] Chan, T.F., Ng, M.K., Yau, A.C., Yip, A.M., 2007. Superresolution image reconstruction using fast inpainting algorithms. Applied and Computational Harmonic Analysis 23, 3–24. doi:10.1016/j.acha.2006.09.005.

[9] Chang, H., Yeung, D.Y., Xiong, Y., 2004. Super-resolution through neighbor embedding, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 275–282. doi:10.1109/CVPR.2004.1315043.

[10] Chang, K., Ding, P.L.K., Li, B., 2018. Single image super-resolution using collaborative representation and non-local self-similarity. Signal Processing 149, 49 – 61. doi:10.1016/j.sigpro.2018.02.031.

[11] Deng, C., Xu, J., Zhang, K., Tao, D., Gao, X., Li, X., 2016a. Similarity constraints-based structured output regression machine: An approach to image super-resolution. IEEE Transactions on Neural Networks and Learning Systems 27, 2472–2485. doi:10.1109/TNNLS.2015.2468069.

[12] Deng, L.J., Guo, W., Huang, T.Z., 2016b. Single image super-resolution by approximated heaviside functions. Information Sciences 348, 107 – 123. doi:10.1016/j.ins.2016.02.015.

[13] Dong, C., Loy, C.C., He, K., Tang, X., 2014. Learning a deep convolutional network for image super-resolution, in: European Conference on Computer Vision (ECCV), pp. 184–199. doi:10.1007/978-3-319-10593-2_13.

[14] Dong, C., Loy, C.C., He, K., Tang, X., 2016. Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 38, 295–307. doi:10.1109/TPAMI.2015.2439281.

[15] Fan, X., Yang, Y., Deng, C., Xu, J., Gao, X., 2018. Compressed multi-scale feature fusion network for single image super-resolution. Signal Processing 146, 50 – 60. doi:10.1016/j.sigpro.2017.12.017.

[16] Fattal, R., 2007. Image upsampling via imposed edge statistics, in: ACM Transactions on Graphics (TOG), p. 95. doi:10.1145/1275808.1276496.

[17] Glasner, D., Bagon, S., Irani, M., 2009. Super-resolution from a single image, in: IEEE International Conference on Computer Vision (ICCV), pp. 349–356. doi:10.1109/ICCV.2009.5459271.

[18] Gong, W., Hu, L., Li, J., Li, W., 2015. Combining sparse representation and local rank constraint for single image super resolution. Information Sciences 325, 1–19. doi:10.1016/j.ins.2015.07.004.

[19] Heo, J.P., Lee, Y., He, J., Chang, S.F., Yoon, S.E., 2012. Spherical hashing, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2957–2964. doi:10.1109/CVPR.2012.6248024.

[20] Hou, H., Andrews, H., 1978. Cubic splines for image interpolation and digital filtering. IEEE Transactions on Acoustics, Speech, and Signal Processing 26, 508–517. doi:10.1109/TASSP.1978.1163154.

[21] Hu, J., Wu, X., Zhou, J., 2018. Noise robust single image super-resolution using a multiscale image pyramid. Signal Processing 148, 157 – 171. doi:10.1016/j.sigpro.2018.02.020.

[22] Huang, J.B., Singh, A., Ahuja, N., 2015a. Single image super-resolution from transformed self-exemplars, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5197–5206. doi:10.1109/CVPR.2015.7299156.

[23] Huang, J.J., Siu, W.C., 2015. Practical application of random forests for super-resolution imaging, in: IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2161–2164. doi:10.1109/ISCAS.2015.7169108.

[24] Huang, J.J., Siu, W.C., 2017. Learning hierarchical decision trees for single-image super-resolution. IEEE Transactions on Circuits and Systems for Video Technology 27, 937–950. doi:10.1109/TCSVT.2015.2513661.

[25] Huang, J.J., Siu, W.C., Liu, T.R., 2015b. Fast image interpolation via random forests. IEEE Transactions on Image Processing 24, 3232–3245. doi:10.1109/TIP.2015.2440751.

[26] Huang, P., Li, T., Shu, Z., Gao, G., Yang, G., Qian, C., 2018a. Locality-regularized linear regression discriminant analysis for feature extraction. Information Sciences 429, 164 – 176. doi:10.1016/j.ins.2017.11.001.

[27] Huang, S., Sun, J., Yang, Y., Fang, Y., Lin, P., Que, Y., 2018b. Robust single-image super-resolution based on adaptive edge-preserving smoothing regularization. IEEE Transactions on Image Processing 27, 2650–2663. doi:10.1109/TIP.2018.2809472.

[28] Hung, K.W., Siu, W.C., 2012. Single image super-resolution using iterative wiener filter, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1269–1272. doi:10.1109/ICASSP.2012.6288120.

[29] Jiang, J., Chen, C., Huang, K., Cai, Z., Hu, R., 2016. Noise robust position-patch based face super-resolution via tikhonov regularized neighbor representation. Information Sciences 367-368, 354 – 372. doi:10.1016/j.ins.2016.05.032.

[30] Johnson, J., Alahi, A., Fei-Fei, L., 2016. Perceptual losses for real-time style transfer and super-resolution, in: European Conference on Computer Vision (ECCV), pp. 694–711. doi:10.1007/978-3-319-46475-6_43.

[31] Kang, S., Kang, P., 2018. Locally linear ensemble for regression. Information Sciences 432, 199 – 209. doi:10.1016/j.ins.2017.12.022.

[32] Keys, R., 1981. Cubic convolution interpolation for digital image processing. IEEE Transactions on Acoustics, Speech, and Signal Processing 29, 1153–1160. doi:10.1109/TASSP.1981.1163711.

[33] Kim, J., Kwon Lee, J., Mu Lee, K., 2016a. Accurate image super-resolution using very deep convolutional networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1646–1654. doi:10.1109/CVPR.2016.182.

[34] Kim, J., Kwon Lee, J., Mu Lee, K., 2016b. Deeply-recursive convolutional network for image super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1637–1645. doi:10.1109/CVPR.2016.181.

[35] Lai, J.Y., Wang, S.L., Liew, A.W.C., Shi, X.J., 2016. Visual speaker identification and authentication by joint spatiotemporal sparse coding and hierarchical pooling. Information Sciences 373, 219 – 232. doi:10.1016/j.ins.2016.09.015.

[36] Lai, W.S., Huang, J.B., Ahuja, N., Yang, M.H., 2017. Deep laplacian pyramid networks for fast and accurate super-resolution, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5835–5843. doi:10.1109/CVPR.2017.618.

[37] Ledig, C., Theis, L., Huszr, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., Shi, W., 2017. Photo-realistic single image super-resolution using a generative adversarial network, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 105–114. doi:10.1109/CVPR.2017.19.

[38] Lehmann, T.M., Gonner, C., Spitzer, K., 2001. Addendum: B-spline interpolation in medical image processing. IEEE Transactions on Medical Imaging 20, 660–665. doi:10.1109/42.932749.

[39] Leng, B., Liu, Y., Yu, K., Zhang, X., Xiong, Z., 2016. 3d object understanding with 3d convolutional neural networks. Information Sciences 366, 188 – 201. doi:10.1016/j.ins.2015.08.007.

[40] Li, J., Gong, W., Li, W., 2015. Dual-sparsity regularized sparse representation for single image super-resolution. Information Sciences 298, 257–273. doi:10.1016/j.ins.2014.11.032.

[41] Li, Y., Wang, Y., Li, Y., Jiao, L., Zhang, X., Stolkin, R., 2016. Single image super-resolution reconstruction based on genetic algorithm and regularization prior model. Information Sciences 372, 196–207. doi:10.1016/j.ins.2016.08.049.

[42] Liu, F., Tang, J., Song, Y., Bi, Y., Yang, S., 2016a. Local structure based multi-phase collaborative representation for face recognition with single sample per person. Information Sciences 346-347, 198 – 215. doi:10.1016/j.ins.2016.02.001.

[43] Liu, L., Peng, Y., Wang, S., Liu, M., Huang, Z., 2016b. Complex activity recognition using time series pattern dictionary learned from ubiquitous sensors. Information Sciences 340-341, 41 – 57. doi:10.1016/j.ins.2016.01.020.

[44] Nayak, R., Patra, D., 2018. New single-image super-resolution reconstruction using mrf model. Neurocomputing 293, 108 – 129. doi:10.1016/j.neucom.2018.02.090.

[45] Oh, B.S., Toh, K.A., Teoh, A.B.J., Lin, Z., 2018. An analytic gabor feedforward network for single-sample and pose-invariant face recognition. IEEE Transactions on Image Processing 27, 2791–2805. doi:10.1109/TIP.2018.2809040.

[46] Peleg, T., Elad, M., 2014. A statistical prediction model based on sparse representations for single image super-resolution. IEEE Transactions on Image Processing 23, 2569–2582. doi:10.1109/TIP.2014.2305844.

[47] Pérez-Pellitero, E., Salvador, J., Ruiz-Hidalgo, J., Rosenhahn, B., 2016a. Antipodally invariant metrics for fast regression-based super-resolution. IEEE Transactions on Image Processing 25, 2456–2468. doi:10.1109/TIP.2016.2549362.

[48] Pérez-Pellitero, E., Salvador, J., Ruiz-Hidalgo, J., Rosenhahn, B., 2016b. Half hypersphere confinement for piecewise linear regression, in: IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1–9. doi:10.1109/WACV.2016.7477651.

[49] Petrosian, A., 2002. New classes of hybrid hadamard-wavelet transforms for signal-image processing, in: Engineering in Medicine and Biology, 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society EMBS/BMES Conference, pp. 153–154. doi:10.1109/IEMBS.2002.1134432.

[50] Pratt, W.K., Kane, J., Andrews, H.C., 1969. Hadamard transform image coding. Proceedings of the IEEE 57, 58–68. doi:10.1109/PROC.1969.6869.

[51] Purkait, P., Pal, N.R., Chanda, B., 2014. A fuzzy-rule-based approach for single frame super resolution. IEEE Transactions on Image Processing 23, 2277–2290. doi:10.1109/TIP.2014.2312289.

[52] Romano, Y., Isidoro, J., Milanfar, P., 2017. Raisr: Rapid and accurate image super resolution. IEEE Transactions on Computational Imaging 3, 110–125. doi:10.1109/TCI.2016.2629284.

[53] Schulter, S., Leistner, C., Bischof, H., 2015. Fast and accurate image upscaling with super-resolution forests, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3791–3799. doi:10.1109/CVPR.2015.7299003.

[54] Shi, J., Liu, X., Zong, Y., Qi, C., Zhao, G., 2018. Hallucinating face image by regularization models in high-resolution feature space. IEEE Transactions on Image Processing 27, 2980–2995. doi:10.1109/TIP.2018.2813163.

[55] Sun, J., Xu, Z., Shum, H.Y., 2008. Image super-resolution using gradient profile prior, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8. doi:10.1109/CVPR.2008.4587659.

[56] Tang, Y., Gong, W., Yi, Q., Li, W., 2018. Combining sparse coding with structured output regression machine for single image super-resolution. Information Sciences 430-431, 577 – 598. doi:10.1016/j.ins.2017.12.001.

[57] Timofte, R., Agustsson, E., Van Gool, L., Yang, M.H., Zhang, L., Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M., et al., 2017. Ntire 2017 challenge on single image super-resolution: Methods and results, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1110–1121. doi:10.1109/CVPRW.2017.149.

[58] Timofte, R., De Smet, V., Van Gool, L., 2013. Anchored neighborhood regression for fast example-based super-resolution, in: IEEE International Conference on Computer Vision (ICCV), pp. 1920–1927. doi:10.1109/ICCV.2013.241.

[59] Timofte, R., De Smet, V., Van Gool, L., 2014. A+: Adjusted anchored neighborhood regression for fast super-resolution, in: Asian Conference on Computer Vision (ACCV), pp. 111–126. doi:10.1007/978-3-319-16817-3_8.

[60] Wang, H., Gao, X., Zhang, K., Li, J., 2017. Single image super-resolution using gaussian process regression with dictionary-based sampling and student- t likelihood. IEEE Transactions on Image Processing 26, 3556–3568. doi:10.1109/TIP.2017.2700725.

[61] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P., 2004. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 13, 600–612. doi:10.1109/TIP.2003.819861.

[62] Xiong, D., Gui, Q., Hou, W., Ding, M., 2018. Gradient boosting for single image super-resolution. Information Sciences 454-455, 328 – 343. doi:10.1016/j.ins.2018.04.075.

[63] Xu, Y., Li, Z., Zhang, B., Yang, J., You, J., 2017. Sample diversity, representation effectiveness and robust dictionary learning for face recognition. Information Sciences 375, 171 – 182. doi:10.1016/j.ins.2016.09.059.

[64] Yang, C.Y., Yang, M.H., 2013. Fast direct super-resolution by simple functions, in: IEEE International Conference on Computer Vision (ICCV), pp. 561–568. doi:10.1109/ICCV.2013.75.

[65] Yang, J., Wright, J., Huang, T.S., Ma, Y., 2010. Image super-resolution via sparse representation. IEEE Transactions on Image Processing 19, 2861–2873. doi:10.1109/TIP.2010.2050625.

[66] Yang, W., Yuan, T., Wang, W., Zhou, F., Liao, Q., 2017. Single-image super-resolution by subdictionary coding and kernel regression. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47, 2478–2488. doi:10.1109/TSMC.2016.2523947.

[67] Ye, M., Ye, H., Yan, G., 2017. HADAMARD Transform Sample Matrix Used in Compressed Sensing Super-Resolution Imaging. Springer International Publishing, Cham. pp. 796–807. doi:10.1007/978-3-319-65298-6_71.

[68] Yu, Z., Wu, F., Yang, Y., Tian, Q., Luo, J., Zhuang, Y., 2014. Discriminative coupled dictionary hashing for fast cross-media retrieval, in: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 395–404. doi:10.1145/2600428.2609563.

[69] Yu, Z., Yu, J., Fan, J., Tao, D., 2017. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering, in: IEEE International

Conference on Computer Vision (ICCV), pp. 1839–1848. doi:10.1109/ICCV.2017.202.

[70] Zeyde, R., Elad, M., Protter, M., 2010. On single image scale-up using sparse-representations, in: International Conference on Curves and Surfaces, pp. 711–730. doi:10.1007/978-3-642-27413-8_47.

[71] Zhang, H., Yang, J., Xie, J., Qian, J., Zhang, B., 2017. Weighted sparse coding regularized nonconvex matrix regression for robust face recognition. Information Sciences 394-395, 1 – 17. doi:10.1016/j.ins.2017.02.020.

[72] Zhang, H., Yang, J., Zhang, Y., Huang, T.S., 2010. Non-local kernel regression for image and video restoration, in: European Conference on Computer Vision (ECCV), pp. 566–579. doi:10.1007/978-3-642-15558-1_41.

[73] Zhang, Y., Fan, Q., Bao, F., Liu, Y., Zhang, C., 2018. Single-image super-resolution based on rational fractal interpolation. IEEE Transactions on Image Processing 27, 3782–3797. doi:10.1109/TIP.2018.2826139.

[74] Zhao, Y., Wang, R., Jia, W., Yang, J., Wang, W., Gao, W., 2018. Local patch encoding-based method for single image super-resolution. Information Sciences 433-434, 292–305. doi:10.1016/j.ins.2017.12.032.

[75] Zhou, F., Li, X., Li, Z., 2018. High-frequency details enhancing densenet for super-resolution. Neurocomputing 290, 34 – 42. doi:10.1016/j.neucom.2018.02.027.

[76] Zhou, Y., Kwong, S., Gao, W., Wang, X., 2016. A phase congruency based patch evaluator for complexity reduction in multi-dictionary based single-image super-resolution. Information Sciences 367-368, 337–353. doi:10.1016/j.ins.2016.05.024.

Algorithm 1 The Algorithm of Clustering and Training

Input: The HR training images I_h and LR training images I_l. The Hadamard matrix Q_{15}. The determined sequence Seq.

Output: The learned SR decision tree dt and the mapping models M.

Extract LR-HR patch pairs from I_h and I_l and initiate the root node with all the training data. Initiate the number of leaf nodes q with 1.
for i = 1; i $\leq$ length(Seq); i++ **do**
    for each unprocessed non-leaf node j **do**
        if $N_j < 3 * min\_num$ **then**
            Mark this node as a leaf node and learn the mapping model M_q. The index q is stored in this leaf node; q++.
        else
            Generate Hadamard patterns P_j for this node as (7).
            Sort the $Seq[i]^{th}$ column of P_j in ascending order and find threshold values v_1 and v_2 as (8, 9).
            Split the training data at node j into its left, middle and right child node by comparing the learned threshold values (v_1 and v_2) and the generated Hadamard patterns P_j.
            if $min(N_l,N_m,N_r)<min\_num$ **then**
                Mark this node as a leaf node and learn the mapping model M_q. The index q is stored in this leaf node; q++.
            else
                Mark this node as a non-leaf node and store the set of parameters β_j in this node.
            end if
        end if
    end for
end for
for each unprocessed non-leaf node j **do**
    Mark this node as a leaf node and learn the mapping model M_q. The index q is stored in this leaf node; q++.
end for

(a) Ground Truth (b) Bicubic (c) ANR[58]

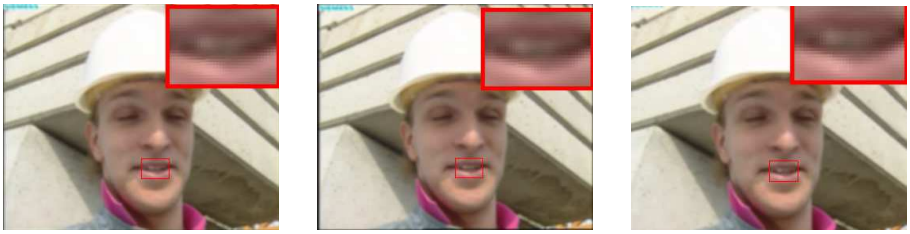
(d) A+[59] (e) SRHDT_f*[24] (f) RFL[53]

(g) SRCNN[13] (h) LapSRN[36] (i) OUR

Figure 8: Comparison of super-resolution on butterfly by different methods for scaling factor 2. The proposed method can better preserve sharp edges.

(a) Ground Truth (b) Bicubic (c) ANR[58]

(d) A+[59] (e) SRHDT_f*[24] (f) RFL[53]



(g) SRCNN[13] (h) LapSRN[36] (i) OUR

Figure 9: Comparison of super-resolution on foreman by different methods for scaling factor 2. The proposed method obtains highest PSNR and can better restore more detailed information.